IMPLEMENTATION AND APPLICATIONS OF OPEN SOUND CONTROL TIMESTAMPS

Andy Schmeder *Adrian Freed*
Center for New Music and Audio Technologies
Department of Music
UC Berkeley
1750 Arch Street, Berkeley CA USA 94720
+1 510 643 9990
{andy,adrian}@cnmat.berkeley.edu

ABSTRACT

The background, purpose, and function of Open Sound Control (OSC) timestamps is reviewed. An analysis shows that jitter-induced noise with dispersion over the millisecond range significantly degrades real-time high-resolution sensor signal streams. The design of a distributed clock synchronization and event scheduling domain over an asynchronous network is described. A realization of this model is presented, created using the new micro-OSC (μOSC) hardware platform and host software components in MaxMSP. An OSC address schema for client-server clock synchronization is documented. Two new objects for MaxMSP are introduced: OSC-timetag and OSC-schedule.

1. INTRODUCTION

Since its introduction in 1997, the Open Sound Control (OSC) protocol [http://opensoundcontrol.org] has been successfully integrated into dozens of hardware and software designs and used in thousands of new music performances and installations. A quality of OSC as a protocol is its design philosophy, which could be paraphrased as "by musicians, for musicians"—many of its features arose from frustrations with the state of the art of digital instrument connectivity including limitations such as slow transmission speed, low-bit depth encodings, and clumsy customization mechanisms.

The use of sensors in computer-musical instruments requires the ability to transmit, interact with, record and analyze time-sampled information at bandwidth and quality sufficient to capture the full-range of expressivity for human-scale gestures. OSC features that support this requirement include high-resolution data formats (e.g. floating point numbers), a high-resolution timestamp format (based on the NTP fixed-point representation [http://ntp.org]), and chronological discrete-event stream semantics attached to OSC *#bundle* headers.

A recent effort by the authors is the cultivation of an online database of OSC implementations and their capabilities [http://opensoundcontrol.org/implementations]. In spite of general consensus that robust temporal information is crucial for musical interfaces [7], the key mechanism for temporal control in OSC, timestamps, has remained elusive—incomplete or unavailable in the majority of implementations. Explanations for the shortcomings in timestamp support by OSC implementations was explored by Freed [3], including programming challenges, data-format and resolution issues, poor operating system support for deadline scheduling, unresolved issues related to synchronization semantics of nested bundles, and the lack of a reference implementation to test functional models against.

2. MOTIVATION AND TOOLS

The event visualization in Figure 1 shows a typical input delay-distribution to a personal computer from an attached human-input device, logged over a 250 millisecond time window. This illustrates the potential magnitude of network-induced delay variability over the millisecond time-scale under common conditions. The problem of delay variability is not restricted to asynchronous transports—even in sample-locked isochronous transports a non-trivial delay distribution exists. Delay and randomness are both inevitable—the task is to manage them. For the realtime transport of OSC packets over asynchronous networks, it is possible to coordinate distributed streams by statistically measuring and anticipating delay, a model for which is described in this paper.

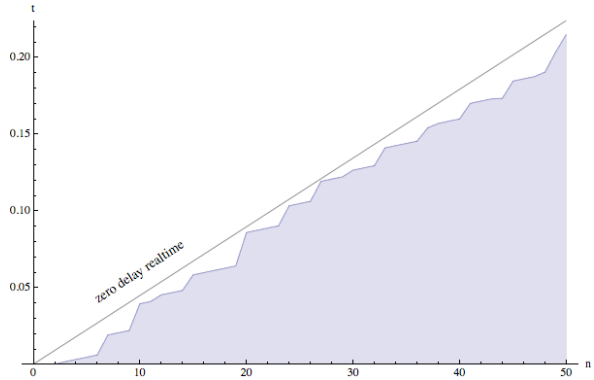

Figure 1: Packet arrival time compared to zero-delay.

A functional implementation of the model is based around the recent development of an OSC-enabled open source firmware for an embedded microprocessor, called micro-OSC (μOSC) [6] (Pictured, Figure 2, the supported Sparkfun "Bitwacker" [http://sparkfun.com]). μOSC is designed to remain as small and compact as

possible while also supporting evolving trends in sensor interfaces such as regulated 3.3 Volt high-resolution sensors, mixed analog and digital multi-rate sensor interfacing, and n > 8-bit data formats [4].

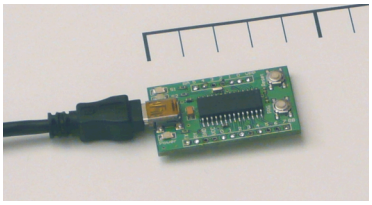

Figure 2: µOSC, an embedded microprocessor gesture-acquisition hardware platform

µOSC exposes low-level microprocessor and hardware controls to a host with a compact and user-friendly OSC address schema. The class-compliant USB-Serial interface conveniently delivers power and data speeds of at least 1.0 Mbits per second, enabling full OSC bundles to be used as the transport wire format. The use of SLIP framing protocol [http://tools.ietf.org/html/rfc1055] gives OSC the ability to be used on any reliable serial transport.

3. JITTER-INDUCED PHASE-MODULATION DISTORTION AT GESTURE-SIGNAL RATES

The implications of phase-modulation distortion are well known in the audio engineering community where picosecond clock dispersion can have a measurable impact on 24-bit/96kHz converters. It is useful to analyze this type of noise in the gesture-frequency domain as well, which we specify over a range of 0-250 Hz in this discussion. In the time domain this corresponds to a sampling rate of 500 Hz, a speed easily realized by µOSC with current hardware capacity. A simulation of the effect of uniform-distributed bounded time jitter shows that effective dynamic range is reduced by phase-modulation noise at levels where it significantly degrades the potential bit-depth of the sensor channel. A representative illustration of the situation with a nominal 10 Hz carrier signal is given in Figure 3.

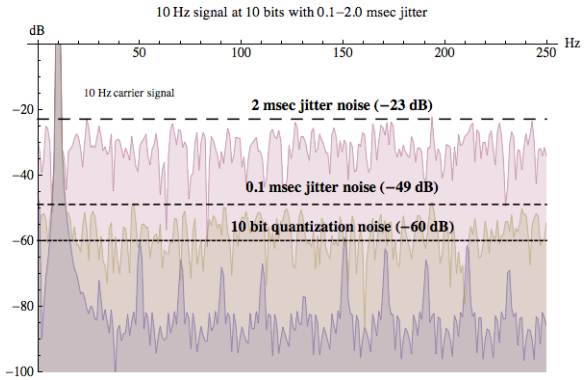

Figure 3: Spectra of jitter and quantization noise

	0.001 msec	0.01 msec	0.1 msec	1. msec	2. msec	4. msec
0.5 Hz	116.088	95.8461	75.8765	55.9886	49.7114	**43.932**
1. Hz	109.916	90.0737	69.6975	49.9082	**43.889**	**37.8536**
2 Hz	103.742	83.9571	63.9379	**43.69**	**38.0029**	**31.7904**
4 Hz	98.0172	77.8748	57.8311	**37.651**	**31.8181**	**25.75**
8 Hz	92.0265	71.7044	51.8118	**31.7585**	**25.8776**	**19.8947**
16 Hz	85.6528	65.8333	**45.7143**	**25.7527**	**20.017**	**13.7036**
32 Hz	79.9328	59.6605	**39.75**	**19.6791**	**13.7678**	**7.75541**
64 Hz	73.7671	53.6596	**33.7997**	**13.6598**	**7.70167**	**1.24085**
128 Hz	67.7876	**47.599**	**27.6065**	**7.54078**	**1.24632**	**3.76947**
256 Hz	61.7541	**41.5826**	**21.6498**	**1.20354**	**3.7653**	**3.73567**

Figure 4: Maximum headroom in dB for tabulated combinations of signal frequency (Hz) and delay dispersion (msec) (simulated, uniformly-distributed, integrated over 10 seconds). **Bold** where headroom < 8-bits.

Timing dispersions on the order tabulated in Figure 4 may arise in contexts such as soft-realtime process control and network datagram transport. What is perhaps under-appreciated is that even at moderate gesture-range frequencies, the sampling-time precision necessary to capture a performance with high-accuracy is well into the sub-millisecond range.

4. TRANSPORT SYNCHRONIZATION FOR GESTURE-SIGNAL STREAMS

Suppose that devices on a network are clock-synchronized (e.g. using NTP, IEEE1588 or another method). Given a reliable clock, a processor can annotate the time-of-occurance for events, and queue events/data for future evaluation. These two functions make possible distributed coordination of signals. Two types of synchronization (causal and anti-causal) are defined for use in a point-to-point network.

4.1. Forward Synchronization

A host transmitting data can synchronize to a future point by anticipation of the one-way forward transport delay. This is called *forward synchronization* [1]. It is useful in signal distribution networks having a star topology with the signal sources at the hub. The order statistic of the maximum transmission delay is a robust non-parametric technique for setting the forward time increment.

4.2. Backward Synchronization

Output of any causal system is not admissible to forward synchronization. Here we consider delay behind the time-of-occurance and make adjustments so that the delay variance is factored out and a constant latency results. The order statistic of the maximum input delay is useful for tuning the rescheduling goal. We call this *backward synchronization*. A visualization of this algorithm in action is depicted in Figure 5.

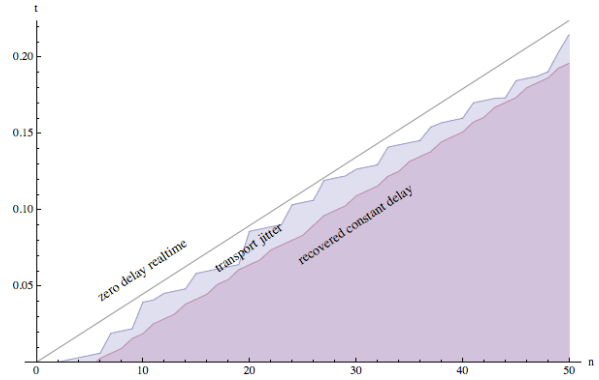

Figure 5: Packet evaluation time corrected to constant-delay after backward synchronization.

5. SYNCHRONIZATION ON MICRO-OSC

The forward/backward synchronization primitives are practical for implementation on embedded processors, enabling µOSC to support time-stamped signal streams. While more advanced models may include sophisticated modes of synchronization (e.g., ad-hoc, wireless), the mechanism described here yields significant improvements in sensor signal conditioning.

5.1. Client-server clock synchronization

The OSC address schema shown enables a host-side implementation of Cristian's algorithm [2] for client-server clock synchronization. To enable the use of the OSC timestamp format as arguments within messages, µOSC uses the extended type tag, `"t"`, for the following:

`/osc/time/accuracy`
`/osc/time/precision`
: µOSC broadcasts these values to the host as advisory messages for tuning the sync accuracy and precision goals. The system builder may derive these values from hardware datasheets and processor execution schedule.

`/osc/time/scale`
: Set/get the internal scale factor mapping device internal timer to external time.

`/osc/time/set`
: Set the device clock.

`/osc/time/inc`
`/osc/time/dec`
: Apply increment/decrement adjustments by time-interval-valued amounts to the device clock.

5.2. Implementation results

Assuming a symmetric round-trip-delay, the minimum input and output delay from µOSC was estimated to be on the order of 3.0 msec (see Figure 8). The clock is then set (using `/osc/time/set`) to $t_{server} + t_{output\text{-}delay}$. When a stable drift rate estimate is available, the scale parameter is adjusted so that clock adjustment increments are decreased. Continuous updates to the drift rate and offset keep the device clock synchronized to the host clock.

The variables affecting synchronization performance are remote (device) clock quality, clock sync quality, and local clock quality. Conditional sampling based on z-score of the minimum order statistic factors out the variance of the transport. In practice, sub-millisecond clock-sync accuracy is easily achievable between µOSC and a computer running MaxMSP. Figure 6 shows a trace of this process in action, regulating the device clock to within 0.1 msec over 3 minutes.

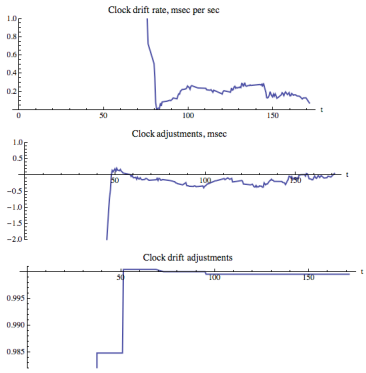

Figure 6: Time-trace of the adaptive clock synchronization in action

The event rescheduling (backward synchronization) part of the design is subject to the precision of the local scheduler precision. The MaxMSP 4.6 scheduler under test exhibited a clock precision of 1.0 msec corresponding to the fastest internal tick-rate, however a conversion directly from OSC-bundles to the MSP signal-domain is possible to obtain the full time-accuracy afforded by the clock synchronization.

6. STATISTICAL CHARACTERIZATION OF DELAY PERFORMANCE

Quality control for real-time behavior of computer-musical instruments and music-gesture protocols has been a concern of past research. A "messy" but reliable testing jig can be constructed for making such measurements in the continuous signal domain, depicted schematically here:

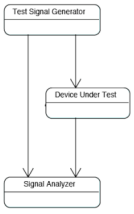

Figure 7: Test jig for latency measurement

This method can be difficult in practice, for example in one case needing special hardware circuits [5] [8], or offline microphone signal processing for peak-estimation of transient physical-events [9]. In a clock-synchronized domain, the use of OSC timestamps

enables greatly simplified measurement of delay-related issues in signal quality control.

Delay in band-limited analog systems is a linear filter, but in digital systems can take on complex behavior when processes such as dynamic queuing and caching strategies are active. The non-parametric minimum and maximum order statistics are useful for establishing expected bounds.

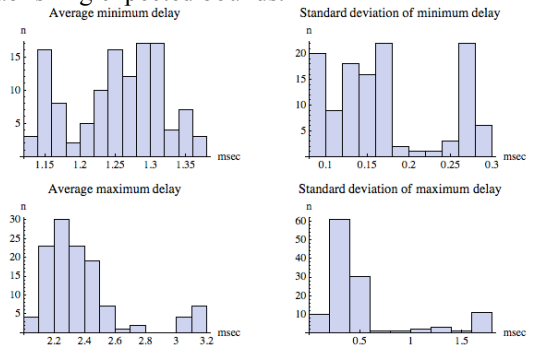

Figure 8: Estimated bounds on input delay for µOSC using USB-Serial transport

As a case study in best-practices for quality control of delay behavior, the synchronized clock on µOSC was used to characterize the USB transport. Histograms of these data are shown in Figure 8. An application of these statistics is in the automatic tuning of the target delay for backward-synchronized event rescheduling.

7. GENERAL PURPOSE TOOLS FOR TIME-AWARE OSC STREAMS IN MAXMSP

The work described here involving time-aware processing is enabled by two new objects for the MaxMSP environment written by the authors that complement the *OpenSoundControl* and *OSC-route* objects by Matt Wright. These object and others made free to the public are published online at http://cnmat.berkeley.edu/downloads.

7.1. OSC-timetag

OSC-timetag provides a microsecond-resolution interface to the system clock using the C standard *gettimeofday()*. It implements mathematical operators on timestamps (difference, scaling, comparison, derivative) and format conversions between OSC/NTP, UNIX time, and ISO8601 strings.

7.2. OSC-schedule

The *OSC-schedule* object buffers OSC bundles in a chronologically sorted priority queue, and outputs them to the closest approximation of event time plus target delay in the MaxMSP scheduler. *OSC-schedule* makes provisions for the user to handle out-of-band packets, e.g. missing the scheduler deadline or having the special "immediate" flag set.

8. CONCLUSION AND FUTURE WORK

The model and implementation shown here establishes a foundation for an end-to-end time-aware computing platform, and shows how phase-modulation noise can be minimized for asynchronous signalling of gesture-range frequencies.

This work could be furthered and improved by exploring more sophisticated clock synchronization methods and hardware, multi-agent network configurations, and delay from block-buffered OSC packet streams (e.g. file streams and database queries). The estimations of signal transmission quality could be refined by the incorporation of input signal density distributions using theoretical prior probabilities and/or gesture-signal statistics.

9. REFERENCES

[1] Brandt, Eli and Dannenberg, Roger, "Time in Distributed Real-Time Systems" in *Proceedings of the ICMC* (San Francisco, CA, USA, 1998) p.523-526

[2] Christian, Flaviu, "Probabilistic clock synchronization" in *Distributed Computing* (Springer Berlin, 1989) p.146-158

[3] Freed, Adrian, "Towards a More Effective OSC Time Tag Scheme", in *Proceedings of the OSC Conference* (Berkeley, CA, USA, June 30 2004)

[4] Freed, Adrian, Avizienis, Rimas and Wright, Matthew, "Beyond 0-5V: Expanding Sensor Integration Architectures" in *Proceedings of the Conference on New Interfaces for Musical Expression* (Paris, France, 2006)

[5] Nelson, Mark and Thom, Belinda, "A Survey of Real-Time MIDI Performance" in *Proceedings of the Conference on New Interfaces for Musical Expression* (Hamamatsu, Japan, 2004)

[6] Schmeder, Andy and Freed, Adrian, "µOSC: The Open Sound Control Reference Platform for Embedded Devices" in *Proceedings of the Conference on New Interfaces for Musical Expression* (Genova, Italy, 2008)

[7] Wessel, David and Wright, Matthew, *Problems and Prospects for Intimate Musical Control of Computers*, Computer Music Journal, Volume 26, Issue 3, 2002, p.11-22

[8] Wright, James and Brandt, Eli, "System-Level MIDI Performance Testing" in *Proceedings of the ICMC* (2001), p.318-321

[9] Wright, Matthew, Cassidy, Ryan J. and Zbyszynski, Michael F., "Audio and Gesture Latency Measurements on Linux and OSX", in *Proceedings of the ICMC* (Miami FL, USA, 2004) p.423-429